http://techies.teamlupus.hu

Vista II: Vista programozás

WPF és WWF

Herczeg Artúr

06



Tartalomjegyzék

Vista programozás I

Bevezető

Ez a tutorial egy kétrészes sorozat első része. A tartalma megyegyezik a *Vista II: Vista programozás* előadássorozaton bemutatott demókkal. A feladatok megoldásához alapvető programozási ismeretekre, illetve a Visual Studio 2005 alapszintű ismeretére van szükség.

Telepítés

Ahhoz, hogy az új fejlesztői funkciók használhatóak legyenek, először a Windows SDK Vistát támogató verziójának telepítése szükséges. Következő lépés Microsoft Visual Studio 2005 telepítése, ha még nem lenne telepítve. Ezután lehet telepíteni a W*F kiterjesztéseket, amelyekkel támogatott módon, a Visual Studio 2005 környezetből lehet fejleszteni.

1. Windows SDK
2. Visual Studio 2005 vagy Visual Studio 2005 Express
3. Visual Studio 2005 Extensions for WF (Windows Workflow Foundation)
4. Visual Studio 2005 Extensions for WCF, WPF (Windows Communication Foundation + Windows Presentation Foundation)

A Windows SDK esetén érdemes meggondolni, hogy mit telepítünk, mivel a telepítés mérete (és a szükséges idő ezzel együtt) 100 MB-tól 1,5GB-ig változik. A kiterjesztések mérete együttesen 10 MB körül mozog.

További információ és a letöltéshez vezető link:
http://msdn.microsoft.com/windowsvista/downloads/products/default.aspx

Előkészületek

Az elkészített feladatok egyszerű kezelése érdekében első lépésként létrehozunk egy Solutiont. A Solution fogja tartalmazni az összes Projectet, amit a feladatok megoldása során megvalósítunk.

1. Kattints a **File/New/Project…** menüpontra
2. Ott válaszd ki az **Other Project Types/Visual Studio Solutions** elemet a baloldalon
3. Válaszd ki a **Blank Solution** projektet a jobboldalon, majd töltsd ki az alábbi mezőket:
    - **Name:** Vista Programming
    - **Location:** *<tetszőleges hely>*
4. Kattints az **OK** gombra

Windows Presentation Foundation

Animáció: Button Glow

A feladat bemutatja, hogy hogyan lehet kialakítani egy egyszerű animációt. A cél egy négyzet megjelenítése, ami 5 másodperc alatt elhalványodik, majd visszatér az eredeti állapotba. Az

animációt egy Play feliratú gomb megnyomásával lehet elindítani, egy Pause feliratú gombbal szüneteltetni, majd egy Resume feliratú gombbal folytatni.

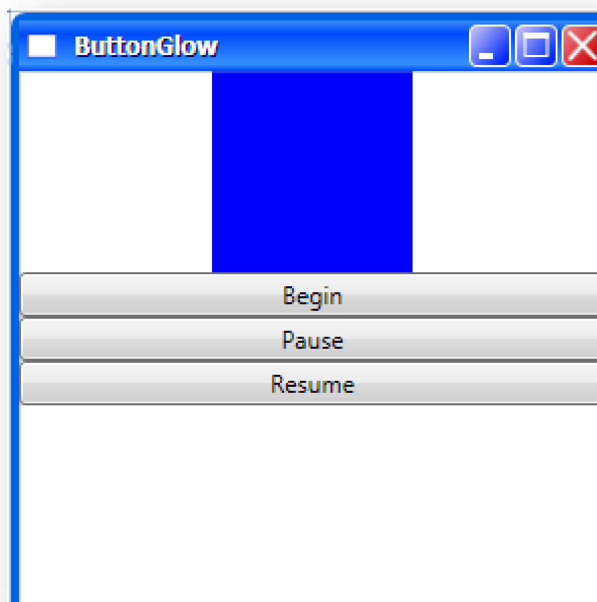
A Project létrehozása

Első lépésként hozzunk létre egy projektet, amiben megvalósíthatjuk az animációt. Ehhez egy Windows Presentation Foundation Application projekt sablont fogunk használni.

1. Kattints a **File/New/Project…** menüpontra
2. Ott válaszd ki a **Visual C#/.NET Framework 3.0** elemet a baloldalon
3. Válaszd ki a **Windows Application (WPF)** projektet a jobboldalon, majd töltsd ki az alábbi mezőket:
    - **Name:** ButtonGlow
    - **Solution:** Add to Solution
4. Kattints az **OK** gombra

A felhasználói felület kialakítása

A következő felületet szeretnék kialakítani kizárólag XAML kód felhasználásával:



Ehhez nyisd meg a **Window1.xaml** fájlt, és módosítsd a tartalmát a következőre:

```
<Window x:Class="ButtonGlow.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="ButtonGlow" Height="300" Width="300"
    >
  <StackPanel>
    <Rectangle Name="rect"
      Width="100"
      Height="100"
      Fill="Blue" />

    <Button Name="btnBegin">Begin</Button>
```

```
    <Button Name="btnPause">Pause</Button>
    <Button Name="btnResume">Resume</Button>
  </StackPanel>
</Window>
```

A ***Rectangle*** elem kék hátterű négyzetet hoz létre. A ***Name*** attribútum segítségével lehet később hivatkozni a négyzetre mind XAML-ből, mind C# kódból.

Az eseményvezérlő struktúra kialakítása

Ahhoz, hogy az animáció elinduljon, valamilyen esemény bekövetkeztéhez kell kötni azt. Esetünkben a ***StackPanel*** eseményeit fogjuk kezelni a ***StackPanel.Triggers*** utasítással. Az ***EventTrigger*** segítségével határozom meg, hogy valamilyen eseményt szeretnék lekezelni, pontosabban egy gombnyomást a ***Begin*** feliratú gombtól. Az ***EventTrigger.Actions*** tartalmazza a tényleges események felsorolását. A ***BeginStoryboard*** egy művelet, ami elindítja az animációt, pontosabban a Storyboard-ot. A Storyboard úgy képzelhető el, mint a Microsoft Interactive Designer-ben a Timeline egymás alatt elhelyezkedő sorai. Ez határozza meg, hogy mit fogunk animálni.

```
<Window x:Class="ButtonGlow.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="ButtonGlow" Height="300" Width="300"
    >
  <StackPanel>
    <Rectangle Name="rect"
      Width="100"
      Height="100"
      Fill="Blue" />

    <Button Name="btnBegin">Begin</Button>
    <Button Name="btnPause">Pause</Button>
    <Button Name="btnResume">Resume</Button>

    <StackPanel.Triggers>
      <EventTrigger RoutedEvent="Button.Click" SourceName="btnBegin">
        <EventTrigger.Actions>
          <BeginStoryboard>

          </BeginStoryboard>
        </EventTrigger.Actions>
      </EventTrigger>
    </StackPanel.Triggers>

  </StackPanel>
</Window>
```

Az animáció lépéseinek meghatározása

Minden alapvető .NET adattípushoz létezik egy animáció. Esetünkben egy double típusú paramétert szeretnénk animálni, így ***DoubleAnimation*** animációra van szükségünk. A ***Storyboard.TargetName*** attribútum határozza meg, hogy melyik objektumot szeretnénk animálni. A ***Storyboard.TargetProperty*** azonosítja a tulajdonságot, aminek az értékét ténylegesen módosítani fogjuk. A ***From*** a kezdőértéket, a ***To*** a végértéket határozza meg.

```
<Window x:Class="ButtonGlow.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
    Title="ButtonGlow" Height="300" Width="300"
    >
  <StackPanel>
    <Rectangle Name="rect"
      Width="100"
      Height="100"
      Fill="Blue" />

    <Button Name="btnBegin">Begin</Button>
    <Button Name="btnPause">Pause</Button>
    <Button Name="btnResume">Resume</Button>

    <StackPanel.Triggers>
      <EventTrigger RoutedEvent="Button.Click" SourceName="btnBegin">
        <EventTrigger.Actions>
          <BeginStoryboard>
            <Storyboard>
              <DoubleAnimation
              Storyboard.TargetName="rect"
              Storyboard.TargetProperty="Opacity"
              From="1.0"
              To="0.1"
              Duration="0:0:5"
              AutoReverse="True"
              RepeatBehavior="Forever" />
            </Storyboard>
          </BeginStoryboard>
        </EventTrigger.Actions>
      </EventTrigger>
    </StackPanel.Triggers>

  </StackPanel>
</Window>
```

Az animáció megállítása, folytatása

Az animáció leállításához létrehozunk két triggert, amit a korábban leírt módon a Pause és Resume gombokhoz kötünk. A ***PauseStoryboard*** és ***ResumeStoryboard*** műveletekkel lehetséges az animációt irányítani.

```
<Window x:Class="ButtonGlow.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="ButtonGlow" Height="300" Width="300"
    >
  <StackPanel>
    <Rectangle Name="rect"
      Width="100"
      Height="100"
      Fill="Blue" />

    <Button Name="btnBegin">Begin</Button>
    <Button Name="btnPause">Pause</Button>
    <Button Name="btnResume">Resume</Button>

    <StackPanel.Triggers>
      <EventTrigger RoutedEvent="Button.Click" SourceName="btnBegin">
        <EventTrigger.Actions>
          <BeginStoryboard Name="story">
```

```
            <Storyboard>
              <DoubleAnimation
              Storyboard.TargetName="rect"
              Storyboard.TargetProperty="Opacity"
              From="1.0"
              To="0.1"
              Duration="0:0:5"
              AutoReverse="True"
              RepeatBehavior="Forever" />
            </Storyboard>
          </BeginStoryboard>
        </EventTrigger.Actions>
      </EventTrigger>
      <EventTrigger RoutedEvent="Button.Click" SourceName="btnPause">
        <EventTrigger.Actions>
          <PauseStoryboard BeginStoryboardName="story" />
        </EventTrigger.Actions>
      </EventTrigger>
      <EventTrigger RoutedEvent="Button.Click" SourceName="btnResume">
        <EventTrigger.Actions>
          <ResumeStoryboard BeginStoryboardName="story" />
        </EventTrigger.Actions>
      </EventTrigger>
    </StackPanel.Triggers>

  </StackPanel>
</Window>
```

Grafikus effektusok: tükröződés

A cél az alább látható tükröződési effektus megvalósítása. Tartalmaz egy képet, amit egy keret ölel körül. A kép mögé árnyék vetül. A kép alatt szögben megtörve a tükröződés.



A project létrehozása

Első lépésként hozzunk létre egy projektet, amiben megvalósítjuk a tükröződést. Ehhez egy Windows Presentation Foundation Application projekt sablont fogunk használni. A létrehozása után beállítjuk, hogy ez legyen az alapértelmezett projekt.

1. Kattints a **File/New/Project…** menüpontra
2. Ott válaszd ki a **Visual C#/.NET Framework 3.0** elemet a baloldalon
3. Váaszd ki a **Windows Application (WPF)** projektet a jobboldalon, majd töltsd ki az alábbi mezőket:

- **Name:** Reflection
- **Solution:** Add to Solution

4. Kattints az **OK** gombra
5. Tedd láthatóvá a **Solution Explorer**-t, ha nem lenne látható
6. Kattints **jobb gombbal a Reflection** projekten
7. Válaszd ki a **Set as Sturtup Project** menüpontot

A kép hozzáadása a projekthez

Az effektus létrehozásához szükség van egy képre, ami vissza fog tükröződni. Így hozzáadjuk a képet a kiválasztott képet a projekthez, hogy könnyebben kezelhető legyen.

1. Tedd láthatóvá a **Solution Explorer**-t, ha nem lenne látható
2. Kattints **jobb gombbal a Reflection** projekten
3. Kattints az **Add/Existing Item…** menüpontra, majd állítsd be a következő mezőt:
   - **Files of type:** Image Files
4. Válassz ki egy tetszőleges képet egy tetszőleges könyvtárból
5. Kattints az **Add** gombra

A kép megjelenítése

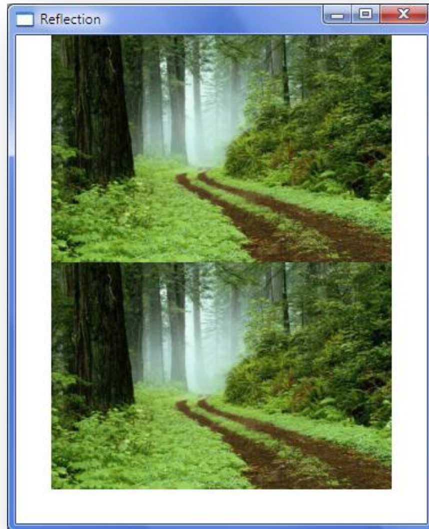


Ahhoz, hogy jól látszódjon az effektus, ***StackPanel*** elrendezézt érdemes használni. Az ***Image*** elem kap egy ***Name*** attribútumot, hogy később hivatkozni lehessen rá. A ***Source*** attribútum határozza meg, hogy honnan kell a képet betölteni. Ezt a legegyszerűbben úgy lehet kitölteni, hogy a Solution Explorerből behúzzuk a képet a két idézőjel közé.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="501"
    >
  <StackPanel>
    <Image
    Name="Forest"
    Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
```

```
    Width="300"
    Height="200"
    Stretch="Fill"/>
  </StackPanel>
</Window>
```

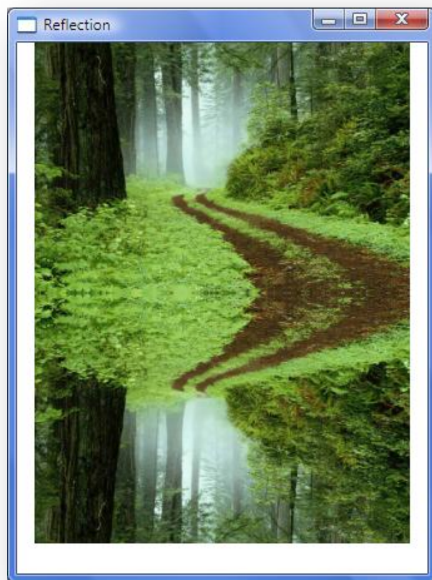
A kép másolatának létrehozása



A tükröződés effektushoz szükségünk van a kép pontos másolatára. Ehhez kiválóan alkalmas a ***VisualBrush***, ami pontosan lemásolja a megadott tartalmat. A másolandó objektumot adatkötéssel kapcsoljuk a VisualBrush-hoz, beállítva a ***Visual*** attribútumot. A VisualBrush egy alapesetben láthatatlan ***Border*** hátterét tölti fel tartalommal.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="412"
    >
  <StackPanel>
    <Image
    Name="Forest"
    Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
    Width="300"
    Height="200"
    Stretch="Fill"/>
    <Border Height="200" Width="300">
      <Border.Background>
        <VisualBrush Visual="{Binding ElementName=Forest}">
        </VisualBrush>
      </Border.Background>
    </Border>
  </StackPanel>
</Window>
```

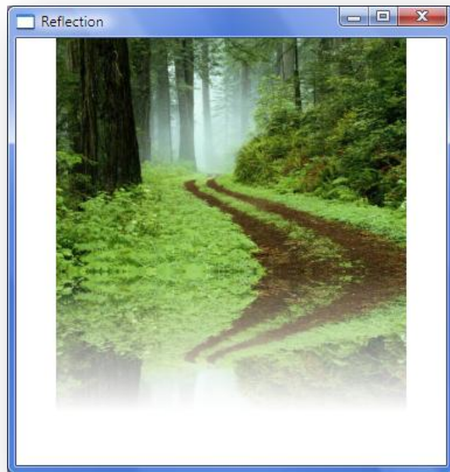
A képmásolat fejtetőre állítása



Ahhoz, hogy a tükröződés illúzióját kapjuk, fejtetőre kell állítani a másolt képet. Ehhez használhatunk forgatást vagy átméretezést. Mi az utóbbit fogjuk alkalmazni. Transzformáljuk a VisualBrush által megjelenített képet, mégpedig úgy, hogy a közepén átfektetett egyenes mentén X irányban megtartjuk az irányokat, Y irányban viszont megfordítjuk.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="412"
    >
  <StackPanel>
    <Image
    Name="Forest"
    Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
    Width="300"
    Height="200"
    Stretch="Fill"/>
    <Border Height="200" Width="300">
      <Border.Background>
        <VisualBrush Visual="{Binding ElementName=Forest}">
          <VisualBrush.Transform>
            <ScaleTransform ScaleX="1" ScaleY="-1" CenterX="300" CenterY="100" />
          </VisualBrush.Transform>
        </VisualBrush>
      </Border.Background>
    </Border>
  </StackPanel>
</Window>
```

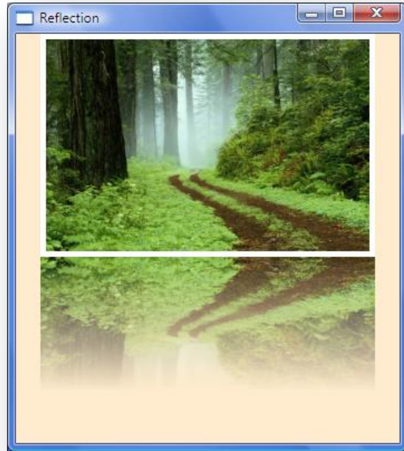
Elhalványodás



A tükröződés jellegzetessége, hogy lefelé halványodik. Ezt úgy érhetjük el, ha lefelé egyre növeljük a képpontok átlátszóságának értékét, amit az ***OpacityMask*** tulajdonsággal tehetünk meg. Ehhez egy színátmenet ecsetet használunk, ami a szürke árnyalatainak segítségével határozza meg az átlátszóságot. 0%-nál fekete színnel teljesen láthatóvá teszi a képet, 60%-nál teljesen eltünteti azt. Ezt úgy teszi, hogy a ***StartPoint*** és az ***EndPoint*** által meghatározott függőleges egyenessel állítja be a színátmenet irányát.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="412"
    >
  <StackPanel>
    <Image
    Name="Forest"
    Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
    Width="300"
    Height="200"
    Stretch="Fill"/>
    <Border Height="200" Width="300">
      <Border.Background>
        <VisualBrush Visual="{Binding ElementName=Forest}">
          <VisualBrush.Transform>
            <ScaleTransform ScaleX="1" ScaleY="-1" CenterX="300" CenterY="100" />
          </VisualBrush.Transform>
        </VisualBrush>
      </Border.Background>
      <Border.OpacityMask>
        <LinearGradientBrush StartPoint="1,0" EndPoint="1,1">
          <GradientStop Offset="0" Color="Black" />
          <GradientStop Offset="0.6" Color="Transparent" />
        </LinearGradientBrush>
      </Border.OpacityMask>
    </Border>
  </StackPanel>
</Window>
```

Keret megjelenítése



Hogy látható legyen a tükröződés határa, érdemes bekeretezni az eredeti képet. Ehhez egy ***Borderrel*** kell körülvenni azt, majd beállítani a keret ecsetének színét és a vonalvastagságot. Hogy a keret jól látszódjon, a háttérszínt átállítjuk fehértől különböző színre.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="412"
    Background="BlanchedAlmond"
    >
  <StackPanel>
    <Border Height="200" Width="300"
            BorderBrush="White" BorderThickness="5"
            HorizontalAlignment="Center"
            VerticalAlignment="Center">
      <Image
      Name="Forest"
      Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
      Width="300"
      Height="200"
      Stretch="Fill"/>
    </Border>
    <Border Height="200" Width="300">
      <Border.Background>
        <VisualBrush Visual="{Binding ElementName=Forest}">
          <VisualBrush.Transform>
            <ScaleTransform ScaleX="1" ScaleY="-1" CenterX="300" CenterY="100" />
          </VisualBrush.Transform>
        </VisualBrush>
      </Border.Background>
      <Border.OpacityMask>
        <LinearGradientBrush StartPoint="1,0" EndPoint="1,1">
          <GradientStop Offset="0" Color="Black" />
          <GradientStop Offset="0.6" Color="Transparent" />
        </LinearGradientBrush>
      </Border.OpacityMask>
```

```
    </Border>
  </StackPanel>
</Window>
```

Árnyékolás vetítése

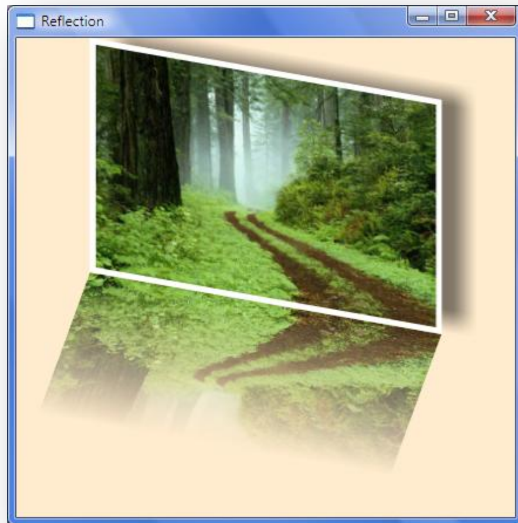


Az árnyékolás effektust a Keretrendszer beépített módon támogatja az alábbi módon.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="412"
    Background="BlanchedAlmond"
    >
  <StackPanel>
    <Border Height="200" Width="300"
            BorderBrush="White" BorderThickness="5"
            HorizontalAlignment="Center"
            VerticalAlignment="Center">
      <Image
      Name="Forest"
      Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
      Width="300"
      Height="200"
      Stretch="Fill"/>
      <Border.BitmapEffect>
        <DropShadowBitmapEffect Color="Black" Direction="20" ShadowDepth="25"
Softness="1" Opacity="0.5" />
      </Border.BitmapEffect>
    </Border>
    <Border Height="200" Width="300">
      <Border.Background>
        <VisualBrush Visual="{Binding ElementName=Forest}">
          <VisualBrush.Transform>
            <ScaleTransform ScaleX="1" ScaleY="-1" CenterX="300" CenterY="100" />
          </VisualBrush.Transform>
        </VisualBrush>
      </Border.Background>
      <Border.OpacityMask>
        <LinearGradientBrush StartPoint="1,0" EndPoint="1,1">
          <GradientStop Offset="0" Color="Black" />
          <GradientStop Offset="0.6" Color="Transparent" />
```

```
        </LinearGradientBrush>
      </Border.OpacityMask>
    </Border>
  </StackPanel>
</Window>
```

A képek térbeli döntése



A képek térbeli döntését a már megjelenített tartalom sarkainak egymáshoz viszonyított eltolásával lehet elérni. Erre a ***RenderTransform*** tulajdonság és a megfelelően paraméterezett ***SkewTransform*** transzformáció alkalmas.

```
<Window x:Class="Reflection.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Reflection" Height="374" Width="412"
    Background="BlanchedAlmond"
    >
  <StackPanel>
    <Border Height="200" Width="300"
            BorderBrush="White" BorderThickness="5"
            HorizontalAlignment="Center"
            VerticalAlignment="Center">
      <Image
      Name="Forest"
      Source="C:\Users\Herczeg Artúr\Documents\Visual Studio 2005\Projects\Vista
Programming\Reflection\Forest.jpg"
      Width="300"
      Height="200"
      Stretch="Fill"/>
      <Border.BitmapEffect>
        <DropShadowBitmapEffect Color="Black" Direction="20" ShadowDepth="25"
Softness="1" Opacity="0.5" />
      </Border.BitmapEffect>
      <Border.RenderTransform>
        <SkewTransform CenterX="0" CenterY="0" AngleX="0" AngleY="10" />
      </Border.RenderTransform>
```

```
    </Border>
    <Border Height="200" Width="300">
      <Border.Background>
        <VisualBrush Visual="{Binding ElementName=Forest}">
          <VisualBrush.Transform>
            <ScaleTransform ScaleX="1" ScaleY="-1" CenterX="300" CenterY="100" />
          </VisualBrush.Transform>
        </VisualBrush>
      </Border.Background>
      <Border.OpacityMask>
        <LinearGradientBrush StartPoint="1,0" EndPoint="1,1">
          <GradientStop Offset="0" Color="Black" />
          <GradientStop Offset="0.6" Color="Transparent" />
        </LinearGradientBrush>
      </Border.OpacityMask>
      <Border.RenderTransform>
        <SkewTransform CenterX="0" CenterY="0" AngleX="-20" AngleY="10" />
      </Border.RenderTransform>
    </Border>
  </StackPanel>
</Window>
```

Windows Workflow Foundation

Egyszerű munkamenet kialakítása

A feladat bemutatja, hogy hogyan lehet elkészíteni egy olyan Workflowt, ami megkapja egy személy adatait, majd a beosztásától függően üdvözli a személyt. Az üdvözlő szöveg a konzolon jelenik meg.

A projekt létrehozása

Első lépésként hozzunk létre egy projektet, amiben megvalósíthatjuk az animációt. Ehhez egy Sequential Workflow Console Application projekt sablont fogunk használni.

1. Kattints a **File/New/Project…** menüpontra
2. Visual C#/Workflow
3. Sequential Workflow Console Application
    - Name: Greetings
    - Solution: Add to Solution
4. Kattints az **OK** gombra

Munkafolyamat változók létrehozása

Ahhoz, hogy a munkafolyamatot kívülről tudjuk irányítani, változókban tárolni kell az éppen feldolgozott információt. Ebben az esetben a munkafolyamat során a személy nevét (Name), beosztását (Position) és az üdvözlő szöveget (Greeting) tároljuk el.

1. Kattints jobb gombbal a **Workflow1.cs** fájlon
2. Válaszd ki a **View Code** menüpontot
3. A **Workflow1** osztályon belül helyezd el az alábbi adattagokat:

```
string name;
string position;
string greeting;
```

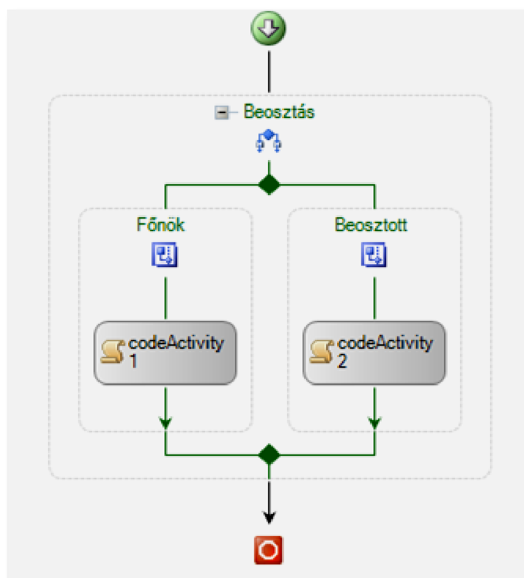
```
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public string Position
        {
            get { return position; }
            set { position = value; }
        }

        public string Greeting
        {
            get { return greeting; }
        }
```

Döntési eset kialakítása

A beosztástól függően kell beállítani az üdvözlőszöveget, így a munkafolyamat változók függvényében kell azt beállítani. Erre jó az IfElse aktivitás. A két CodeActivity aktivitást a következő lépésként fogjuk elhelyezni a munkamenet ábrán.



1. Kattints jobb gombbal a **Workflow1.cs** fájlon
2. Válaszd ki a **View Designer** menüpontot
3. Tedd láthatóvá a Toolbox-ot
4. Húzz egy **IfElse** aktivitást az ábrára, a Drop Activities to create a Sequential Workflow területre
5. Az **ifElseActivity1 Name** tulajdonságát állítsd át Beosztás értékre
6. Kattints az Beosztás bal ágára
7. Az **ifElseBranchActivity1 Name** tulajdonságát állítsd át Főnök értékre
8. A **Condition** tulajdonságnál válaszd ki a **Declarative Rule Condition** elemet a legördülő listából
9. Nyisd ki a **Condition** tulajdonságot
10. A **ConditionName** tulajdonságot állítsd Főnök értékre
11. Az **Expression** tulajdonság mellett kattints a három ponttal jelzett gombra

12. Gépeld be a következő kódot: `this.position == "Főnök"`
13. Kattints az Beosztás jobb ágára
14. Az **ifElseBranchActivity2 Name** tulajdonságát állítsd át Beosztott értékre
15. A **Condition** tulajdonságnál válaszd ki a **Declarative Rule Condition** elemet a legördülő listából
16. Nyisd ki a **Condition** tulajdonságot
17. A **ConditionName** tulajdonságot állítsd Beosztott értékre
18. Az **Expression** tulajdonság mellett kattints a három ponttal jelzett gombra
19. Gépeld be a következő kódot: `this.position == "Beosztott"`

Üdvözlőszöveg beállítása

Most elhelyezzük a két CodeActivity aktivitást az ábrán. A feladatuk az üdvözlő szöveg beállítása lesz.

1. Tedd láthatóvá a Toolbox-ot
2. Húzz egy-egy **Code** aktivitást a Drop Activities Here feliratokra
3. Válaszd ki a **codeActivity1** aktivitást
4. Az **ExecuteCode** tulajdonságot állítsd be `execute_fonok` értékre
5. Az így létrehozott eljárásba írd be a következő kódot:
   ```
   greeting = string.Format("Üdvözlöm {0}", name);
   ```
6. Válts vissza a **[Design]** nézetre
7. Válaszd ki a **codeActivity2** aktivitást
8. Az **ExecuteCode** tulajdonságot állítsd be `execute_beosztott` értékre
9. Az így létrehozott eljárásba írd be a következő kódot:
   ```
   greeting = string.Format("Hello {0}", name);
   ```

Üdvözlőszöveg kiírása

Már csak ki kell íratnunk a konzolba az üdvözlő szöveget. Ennek érdekében módosítjuk azt az eseménykezelő (anonim) függvényt, ami a munkamenet végrehajtása után fut le. Az e.OutputParameters tömb tartalmazza a kimeneti paramétereket, azaz azokat a tulajdonságokat, melyeknek csak getter (get{}) részük van.

1. Kattints jobb gombbal a **Program.cs** fájlra
2. Válaszd ki a **View Code** menüpontot
3. **Keresd meg** a
   `workflowRuntime.WorkflowCompleted += delegate(object sender, WorkflowCompletedEventArgs e)` delegáltat
4. A `waitHandle.Set();` elé **illeszd be** a következő kódot:
   ```
   Console.WriteLine(e.OutputParameters["Greeting"]);
   Console.ReadKey();
   ```

A munkafolyamat paraméterezése

Ahhoz, hogy a munkamenet kívülről is állítható legyen, be kell állítani a munkamenet változók értékét. Ehhez paraméterként átadjuk a változók kezdeti értékét.

1. **Keresd meg** a
   `WorkflowInstance instance = workflowRuntime.CreateWorkflow(typeof(WorkflowConsoleApplication1.Workflow1));` sort
2. **Elé illeszd be** a következő kódot:
   ```
   Dictionary<string, object> parameters = new Dictionary<string, object>();
   ```

```
parameters.Add("Name", "Péter");
parameters.Add("Position", "Főnök");
```

3. **Módosítsd a deklarációt** a következő módon:
```
WorkflowInstance instance =
workflowRuntime.CreateWorkflow(typeof(Greetings.Workflow1), parameters);
```

Saját aktivitás létrehozása

A feladat bemutatja, hogy hogyan lehet saját aktivitást létrehozni. Az aktivitás egy Windows üzenetdobozban (MessageBox) fogja megjeleníteni a megadott tartalmat.

A projekt létrehozása

Első lépésként hozzunk létre egy projektet, amiben megvalósíthatjuk az animációt. Ehhez egy Workflow Activity Library projekt sablont fogunk használni.

5. Kattints a **File/New/Project…** menüpontra
6. Visual C#/Workflow
7. Workflow Activity Library
   - Name: MessageBox
   - Solution: Add to Solution
8. Kattints az **OK** gombra

Aktivitás változók deklarálása

Ahhoz, hogy az aktivitás tetszőleges tartalmat tudjon megjeleníteni az üzenetdobozban, a tartalmat át kell adni az aktivitásnak. Azért, hogy az aktivitás munkamenet egyéb változóitól könnyen, listából választható módon kaphasson tartalmat, érdemes a **DependencyProperty** típust alkalmazni paraméterként.

1. Kattints jobb gombbal az **Activity1.cs** fájlon
2. Válaszd ki a **View Code** menüpontot
3. Az Activity1 osztályon belülre **illeszd be** a következő kódot:
```
public static DependencyProperty MessageProperty =
DependencyProperty.Register(
            "Message", typeof(string), typeof(MessageBox),
            new PropertyMetadata("Your message here"));
public string Message
{
     get
     {
         return (string)base.GetValue(MessageProperty);
     }
     set
      {
          base.SetValue(MessageProperty, value);
      }
 }
```

Az üzenetdoboz megjelenítése

Az üzenetdoboz megjelenítéséhez a szabványos **System.Windows.Forms.MessageBox** osztályt használjuk. Az **Execute** metódus bíráljuk felül, ami az aktivitás utasításainak végrehajtásáért felelős.

1. Kattints jobb gombbal a MessageBox projekten belül a **References** elemen
2. Válaszd ki az **Add Reference…** menüpontot
3. A .NET lapon keresd meg a **System.Windows.Forms** szerelvényt

4. Kattints az **OK** gombra
5. Az Activity1 osztályon belülre **illeszd be** a következő kódot:

```
protected override ActivityExecutionStatus Execute(ActivityExecutionContext
executionContext)
{
      System.Windows.Forms.MessageBox.Show(Message);
      return ActivityExecutionStatus.Closed;
}
```

6. Code nézetben kattints jobb gombbal az **Activity1 szövegrészen**
7. Válaszd ki a **Refactor/Rename…** menüpontot
8. A **New name** mező tartalma legyen MessageBox
9. Kattints az **OK** gombra
10. Kattints az **Apply** gombra

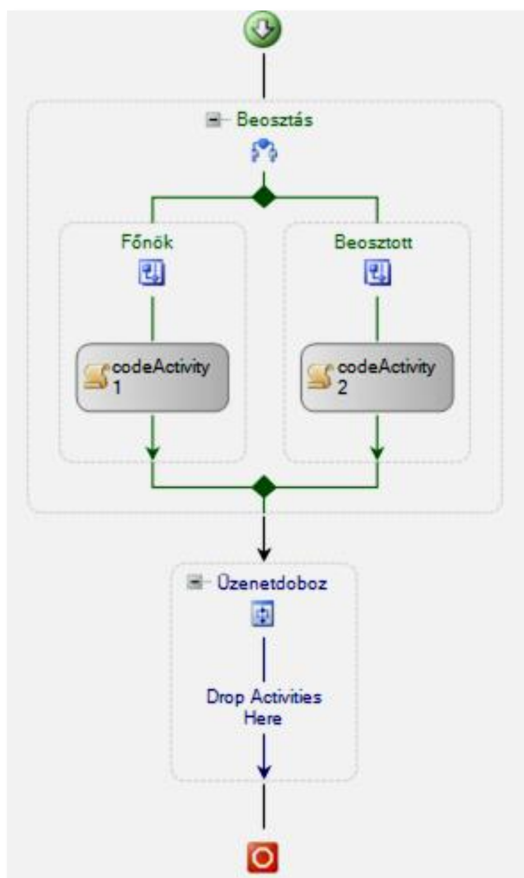
A projekt lefordítása

Ahhoz, hogy később dolgozni tudjunk az új aktivitással, le kell azt fordítani.

1. Válaszd ki a **Build/Build MessageBox** menüpontot

Saját aktivitás használata

A feladat bemutatja, hogy hogyan lehet a korábban létrehozott saját aktivitást egy munkamenetben felhasználni.



1. Kattints a Greetings projekt **Workflow1.cs** elemém jobb gombbal
2. Válaszd ki a **View Designer** menüpontot
3. Tedd láthatóvá a Toolbox-ot

4. Húzz egy **MessageBox** aktivitást az elágazás aktivitás után
5. Állítsd a **Name** tulajdonságot Üzenetdoboz értékre
6. Kattints a **Message** tulajdonság mellett a **…** feliratú gombra
7. Válaszd ki a **Greeting** tulajdonságot
8. Kattints az **OK** gombra

A szerzőről

Készítette:	Herczeg Artúr
Blog:	http://techies.teamlupus.hu
Email:	artur.herczeg(kukac)studentconsultant.org